

Nika

PROJECT: GEAR5

Group 10

Team Leader:

Ralph Robes, N01410324

Members:

Francisco Santos, N01423860

Elijah Tanimowo, N01433560

Pradeep Singh, N00975892

Table Of Contents

Our Project	3
Student Names & IDs	3
Github Repo Link	3
Meaningful Offline Mode Functionality	3
Runtime Permission	3
Technical Debt Addressment	4
Suggestions	4
Sprint	4
Sprint Goals	4
Scrum Dashboard	5
Dashboard 1	5
Dashboard 2	5
Dashboard 3	6
Post-Mortem	7
Overview	7
Performance Review	7
Time Usage	7
Quality	8
Lessons	8
Attendance	8
Project Review Meeting	9
C4 Model	10
Container Diagram	10
Component Diagrams	11
Component Diagram 1 (Mobile App)	11
Component Diagram 2 (PCB)	12
Areas of Refactoring	12
Area 1	12
Area 2	15

Our Project

Student Names & IDs

Name	Id	Signature	Effort
Ralph Robes	N01410324	<i>Ralph Robes</i>	100%
Francisco Santos	N01423860	<i>Francisco Santos</i>	100%
Pradeep Singh	N00975892	<i>Pradeep Singh</i>	100%
Elijah Tanimowo	N01433560	<i>Elijah Tanimowo</i>	100%

Github Repo Link

<https://github.com/RalphRobes0324/Nika.git>

Meaningful Offline Mode Functionality

Through the usage of the Gear5 App, you use Bluetooth to connect with our hardware devices, which means that wifi is not a requirement for using our app, and the offline functionality that we support is to successfully update your top score to the database. This means that even if you obtain a new high score while offline, once you go online and enter into the app, your top score will be uploaded to the database and will become your new score.

Runtime Permission

One example of runtime permission that we implemented was our Bluetooth connection. If the user navigates to our BluetoothFragment, which can be found by clicking the overflow menu and then on the “connect” option, the user will be presented with three options. The options are “TURN ON”, “TURN OFF” and “DISCOVERABLE”. If the user selects the “DISCOVERABLE” option, then they are presented with a pop-up asking whether or not they would like to enable Bluetooth. Once that has been done, the user is able to turn on and off Bluetooth with the other two options. In order to make sure the Bluetooth connection was more than a simple pop-up and that it actually had functionality, we included various functions within the java code, as well as including various lines into the AndroidManifest, such as:

```
<uses-permission android:name="android.permission.BLUETOOTH" />
<uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />
<uses-permission android:name="android.permission.BLUETOOTH_ADVERTISE" />
<uses-permission android:name="android.permission.BLUETOOTH_CONNECT" />
<uses-permission android:name="android.permission.BLUETOOTH_SCAN" />
```

Technical Debt Addressment

We addressed our technical debt by returning to our code if needed between each deliverable. We would prioritize the completion of the objective and if needed, we would then go back and then change it. One large example of our team addressing our technical debt would be the code used in the profile document. The first iteration of it was messy, and hard to understand and read, and this made it difficult for the team members who weren't the ones who worked on it. As a result of this reasoning, it was addressed and remade to meet all our needs for understandability.

Suggestions

Overall the assignment was alright, as it encompassed many aspects that make an app functional. We really enjoyed being free to add certain fragments instead of them all being set in stone. That being said, there are a few changes that we would suggest. It would have been nice if certain aspects were not mandatory, as we believe we could have had a fully functioning app that can complete all the same tasks as our current app without clutter and unnecessary features. Perhaps in future years, you could make the assignments even more flexible to allow everyone's creativity to really shine. All things considered, this was a difficult, yet fun and enjoyable, experience and we enjoyed creating our app to the fullest.

Sprint

Sprint Goals

- Three types of testings (JUnit4, Robolectric, Espresso)
- Implement Offline Mode functionality
- Implement Personal Score Fragment
- Update Login/Registration requirements
- Update UI design for Consistency

Scrum Dashboard

Dashboard 1

Dashboard 1 displays five columns of tasks, each with a title, a progress bar, and a 'Add a card' button at the bottom.

- Column 1: Email or phone text capabilities in order to send users free currency to incentivize them to use our app more**
 - Task 1: Add a section for the user to add email/phone number and save it to the database
 - Task 2: Create an automated system that sends an email/text on a regular basis i.e. once per month
 - Task 3: Have a one-time email/text with free currency sent to new users (can detect if their phone number/email is already in the database)
 - Task 4: Add system for the user to change their phone number or email address
 - Task 5: Add system for the user to remove the phone or email (database could hold this info for a period of time in case they change their mind and so the free currency emails/text couldn't be abused)
- Column 2: Leaderboard where users can compete for rewards, thus improving their experience**
 - Task 1: Code to create leaderboard with usernames from the database
 - Task 2: Code to display scores collected from the database
 - Task 3: Add code to reset leaderboard on monthly intervals
 - Task 4: Database will find the top 3 users with the highest scores and add currency to their account during the leaderboard reset
 - Task 5: Allow user to opt in/out of leaderboard (if opted out they can't get prizes)
- Column 3: Partner with Twitch/YouTube streamers to allow users to see early footage of gameplay and receive in game bonuses**
 - Task 1: Pay content creators to promote our gaming system.
 - Task 2: Work with Twitch to add drops for free currency/skins
 - Task 3: Make ads short and engaging in order to save money while still generating interest
 - Task 4: Put our ads in gaming-related YouTube content in order to reach our intended audience
 - Task 5: Give streamers codes that they can then send out to their fans
- Column 4: Customization for users, where they can change most aspects of the game for personalization and comfort. Supports users with disabilities by allowing them to move and position controller functionality**
 - Task 1: Add additional coding to allow user to change joystick position
 - Task 2: Add additional coding to allow user to change button position
 - Task 3: Add additional coding to allow user to change background
 - Task 4: User can use additional parts in order to modify their system to their liking
 - Task 5: User can have custom images for Avatar in their profile
- Column 5: Development for users, where they can create their own games in order to earn some money**
 - Task 1: Add fragment for user development
 - Task 2: Add store, where users can buy other user created games
 - Task 3: Add system for user to convert their game currency into their local currency
 - Task 4: Add tools for user to easily create games
 - Task 5: Add creator benefits, where users who create games get additional rewards based on the games sales and popularity

Dashboard 2

Dashboard 2 displays four columns of tasks, each with a title, a progress bar, a due date, and a status icon.

- Column 1: Story 1: Provide a way for users to give feedback and improve user experience**
 - Task 1: implement star rating (Medium) - Due: Oct 28 - Oct 31 - Status: ET
 - Task 2: implement user comment/feedback (Medium) - Due: Oct 29 - Nov 6 - Status: ET
 - Task 3: allow user to provide phone number (Small) - Due: Oct 29 - Nov 4 - Status: ET
 - Task 4: allow user to provide email (Small) - Due: Oct 28 - Oct 31 - Status: ET
 - Task 5: allow user to provide name (Small) - Due: Oct 29 - Oct 31 - Status: ET
- Column 2: Story 2: Save user information to a database so that the user can seamlessly switch devices and allow them to use other account formats**
 - Task 1: Save login info to firebase (Large) - Due: Oct 25 - Nov 2 - Status: RR
 - Task 2: Save review to firebase (Medium) - Due: Nov 11 - Nov 13 - Status: RR
 - Task 3: Save balance to firebase (Medium) - Due: Nov 12 - Nov 13 - Status: RR
 - Task 4: Implement Google sign in (Large) - Due: Oct 28 - Nov 3 - Status: RR
 - Task 5: Allow user to register new account with erasing old one, login and register options and stores to firebase separately (Large) - Due: Nov 2 - Nov 5 - Status: RR
- Column 3: Story 3: Provide user with the ability to customize app to their preferences, thus improving their experience**
 - Task 1: Implement ability to switch profile picture (Small) - Due: Oct 22 - Oct 24 - Status: FS, PS
 - Task 2: Allow user to change joystick colour, using user preferences (Medium) - Due: Oct 31 - Nov 6 - Status: FS
 - Task 3: Allow user to change button colour, using shared preferences (Large) - Due: Oct 24 - Oct 29 - Status: FS
 - Task 4: Allow user to lock the app to portrait (Medium) - Due: Oct 26 - Oct 30 - Status: FS
 - Task 5: Allow user to mute the app (Medium) - Due: Nov 9 - Nov 11 - Status: FS
- Column 4: Story 4: Improve the UI to give the user a better overall experience with the app**
 - Task 1: Change backgrounds to match fragment themes (Small) - Due: Oct 30 - Nov 10 - Status: PS
 - Task 2: Change font style to better represent the app (Small) - Due: Nov 7 - Nov 9 - Status: PS
 - Task 3: Change colours to match the app (Small) - Due: Nov 1 - Nov 3 - Status: PS
 - Task 4: Add screens to add a better flow to the app (Large) - Due: Oct 29 - Nov 10 - Status: ET, FS, PS, RR
 - Task 5: Improve menu navigation to make everything clearer for the user (Medium) - Due: Nov 4 - Nov 6 - Status: PS

Dashboard 3

Story 1: Implement Offline Mode capability for users without access to Internet ...

Task 1: Allow users to play games Via Offline/Bluetooth
Francisco Santos 🧑🏻

Task 2: Users will be able to interact with application in an Offline environment
Francisco Santos 🧑🏻

Task 3: Allow users to continue using application even if connectivity is interrupted
Francisco Santos 🧑🏻

Task 4: Users will be able to have access their score and save
Francisco Santos 🧑🏻

Task 5: Once user is connected via Wifi, score will be checked and updated
Elijah Tanimowo 🧑🏻

Story 2: Added Security for User passwords implementing numeric, special, and upper/lower case letters ...

Task 1: User must have password that contains Numeric, Special, Upper/Lower case characters
Ralph Robes 🧑🏻

Task 2: Dialog appears if the password does not meet requirements
Ralph Robes 🧑🏻

Task 3: Allows user to login via Email or Username
Ralph Robes 🧑🏻

Task 4: Create Forgot password function, so users may reset password
Ralph Robes 🧑🏻

Task 5: Users provided added security through google
Elijah Tanimowo 🧑🏻

Story 3: Allow users to connect with external hardware ...

Task 1: Users will be able to connect with most devices via bluetooth
Pradeep Singh 🧑🏻

Task 2: Allow users to use third party hardware such as headphones, controllers, speakers
Elijah Tanimowo 🧑🏻

Task 3: Users will be able to use controls within the app virtually and physically through joystick implemented on Arcade System
Pradeep Singh 🧑🏻

Task 4: If user disconnects with any hardware, game will be paused with alert
Pradeep Singh 🧑🏻

Task 5: Device saves bluetooth configuration, reducing the need to pair personal devices.
Pradeep Singh 🧑🏻

Post-Mortem

Overview

Category	Lesson Learned	Achieved?	Comments
Project Planning	Product concept was appropriate to Business Objectives	Yes	
	Project Plan and Schedule were well-documented, with appropriate structure and detail	Yes	
	Project Schedule encompassed all aspects of the project	Yes	
	Tasks were defined adequately	Yes	
	Requirements were documented clearly	Yes	
	Specifications were clear and well-documented	Yes	
	Project budget was well defined	Yes	Desired budget of \$100 cad
Project Execution	Project stuck to its original goals	Yes	
	Changes in direction that did occur were of manageable frequency and magnitude	Yes	
	Project baselines (Scope, Time, Cost, Quality) were well-managed (e.g., changed through a formal Change Control Process)	Yes	
	Design changes were well-controlled	Yes	
	The project had adequate Quality Control	Yes	Sudo deadlines created managed team timeline
	Project Manager was effective	Yes	
	Project Team was properly organized and regulated	Yes	Members took accountability for their work and achieved their goals on time
	Project team worked effectively on project goals	Yes	
	There was good communication within the Project Team	Yes	This can be improved on; a well devised plan was taken into place post deliverable 3
	Resources were not over-committed	Yes	
	Changes were consistently committed	Yes	
	Authority and accountability were well defined within the group	Yes	
Overall	Initial cost and schedule estimates were accurate	No	We spent slightly over our estimated goal of \$100 cad total
	Product was delivered within schedule	Yes	
	Product was delivered within budget	No	Slightly over budget, as more costs were added such as Google fee
	Technology chosen was appropriate	Yes	
	The project was a success	Yes	Desired implementations were successful
	Project Objectives were met	Yes	All deadlines and requirements were met

Performance Review

Overall the group work was done well. No project goes without hiccups, but managing to pull through and meet deadlines is what we did. Each member has individual strengths and weaknesses, it is finding out how to implement each member to a key area in the project where we would have to mitigate this. All members took responsibility for their parts and we did not run into issues with work not done. In terms of Cost, we all kept to a reasonable budget and spent slightly lower than we initially expected. In all the group project was a success and we look forward to what next semester will bring us.

Time Usage

Time usage was a mixed bag due to everyone having other classes that they must attend and complete work for. For the most part, time was managed wisely and things were completed in a timely and organized fashion. Some parts, however, had to be completed last second due to a time crunch from upcoming exams and many final assignments that we have been given. With everything said and done, we believe our time usage was managed wisely and to the best of our abilities.

Quality

The overall quality of the app is quite good. There are not any huge glaring issues when it comes to quality, however, some editions will simply have to wait until next semester due to the time constraints that we have.

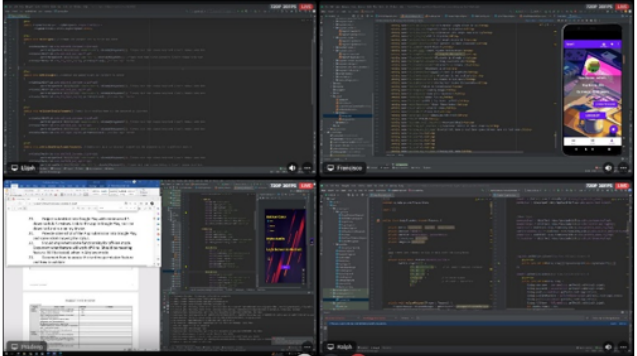
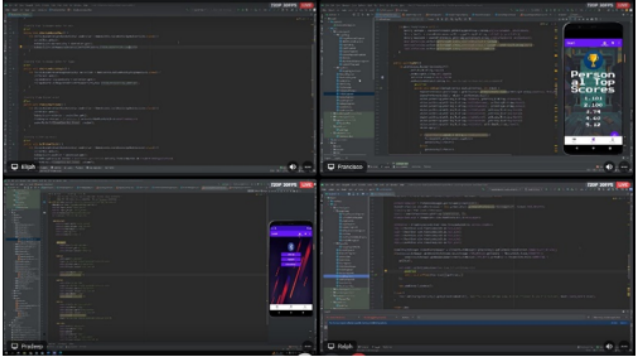
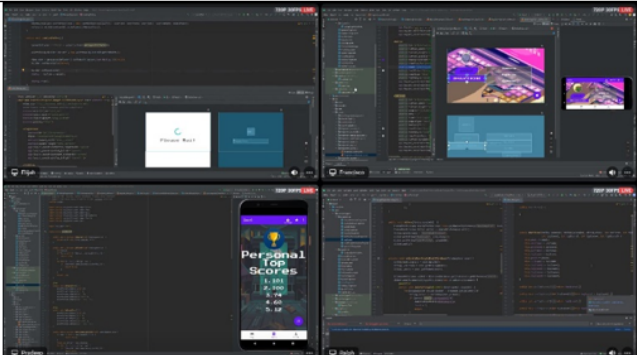
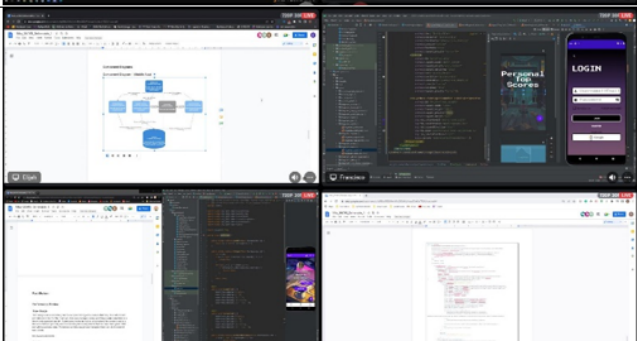
Lessons

We learned that creating a mobile app takes a lot of work, time, and dedication. The app came out well in the end, however, if we were not busy with other classes, projects, and exams, we believe that the app could have turned out even more exceptional. Some mistakes we made in the earlier deliverables did not realize how much time the written portion would actually take up. To that end, we made sure to start our deliverables as soon as possible in order to ensure the completion and high standard that we know you expect.

Attendance

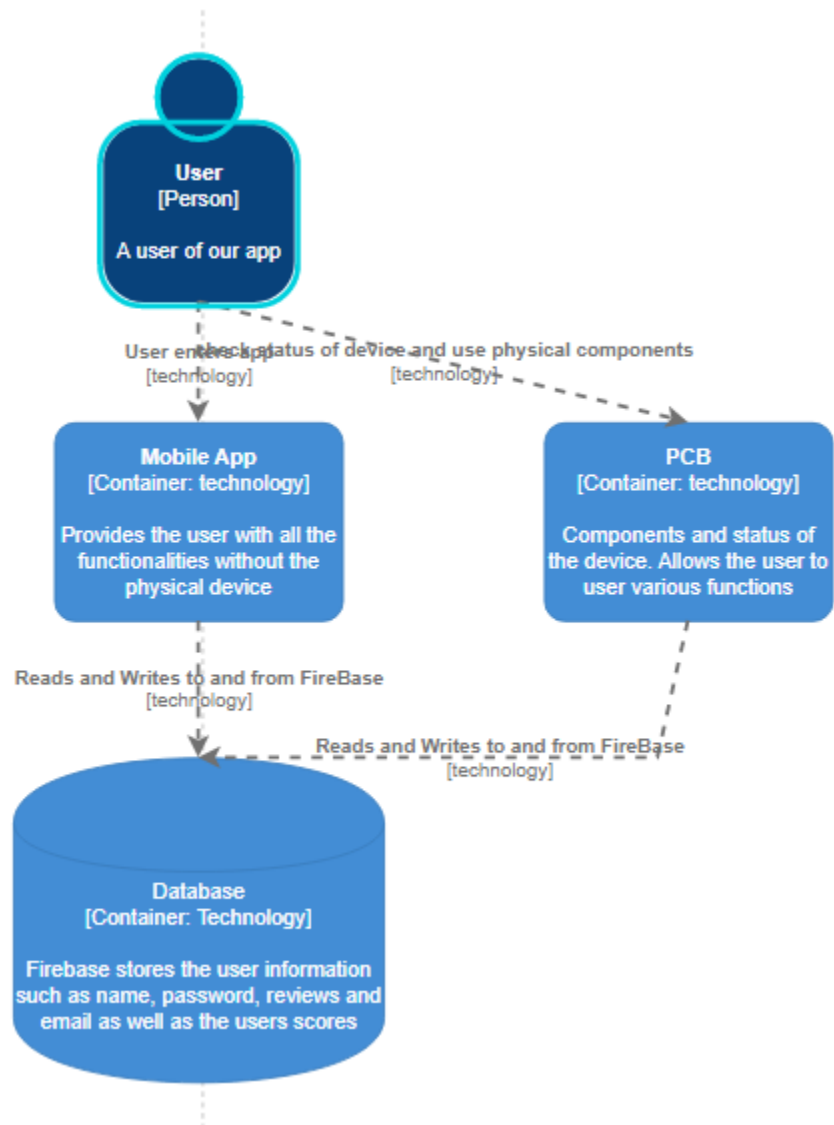
When we conducted our various meetings throughout the course of the project, everyone always made sure to attend. When there was a situation where a member could not make it, we rescheduled. Therefore our attendance for all group members was 100%.

Project Review Meeting

Meeting	Outcome	Screenshot
<p>1</p> <p>November 28th, 2022</p>	<ul style="list-style-type: none"> Allocated Work amongst each member Rebuilt FireBase and Login registration system 	
<p>2</p> <p>November 30th, 2022</p>	<ul style="list-style-type: none"> Added testing for Espresso and roboelectric Added Testing for Junit4 Added implementation for Offline mode Rebuilt FireBase and Login registration system 	
<p>3</p> <p>December 2nd, 2022</p>	<ul style="list-style-type: none"> Implemented Simulation results of receiving score to save into our database Started documentation for Deliverable 4 Added implementation for Offline mode 	
<p>4</p> <p>December 4th, 2022</p>	<ul style="list-style-type: none"> Review of Final project and Application Final touches including uncommenting code and reviewing design principles 	

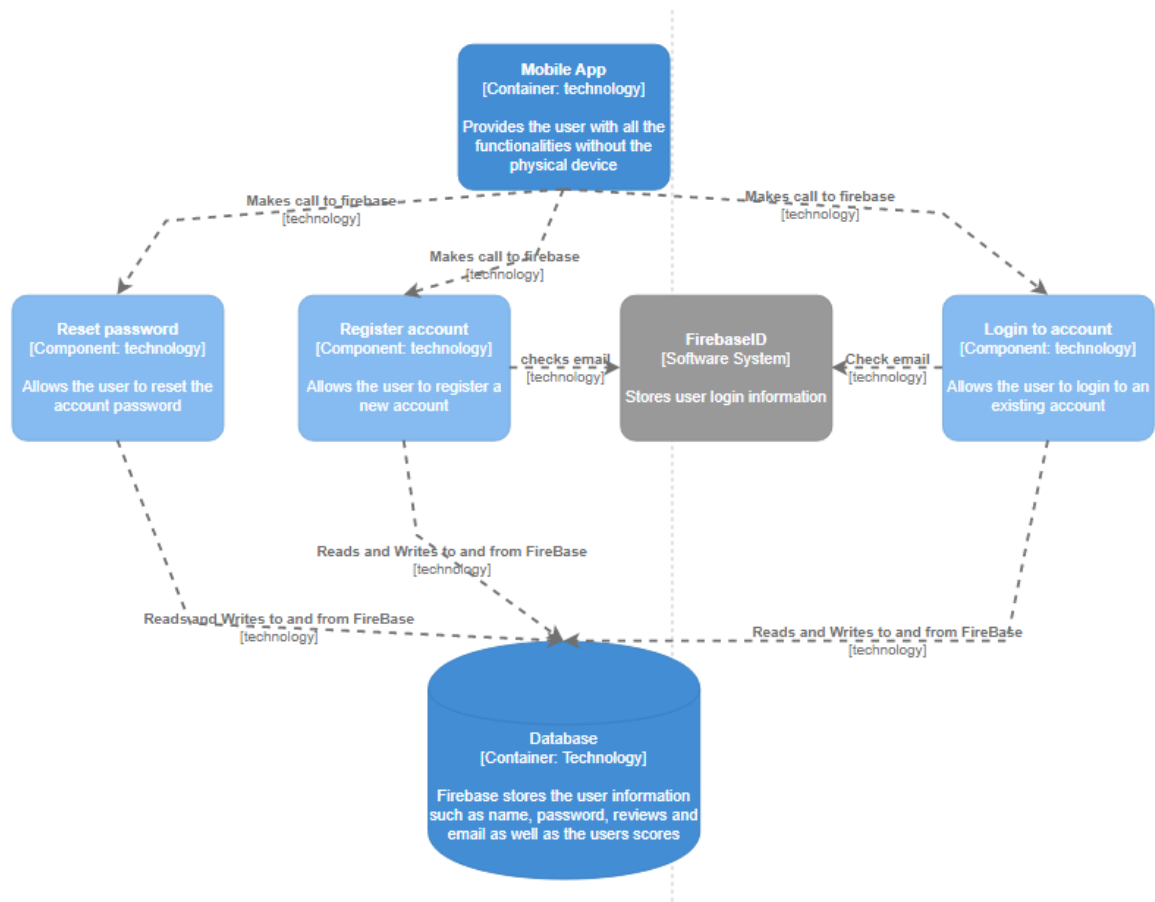
C4 Model

Container Diagram

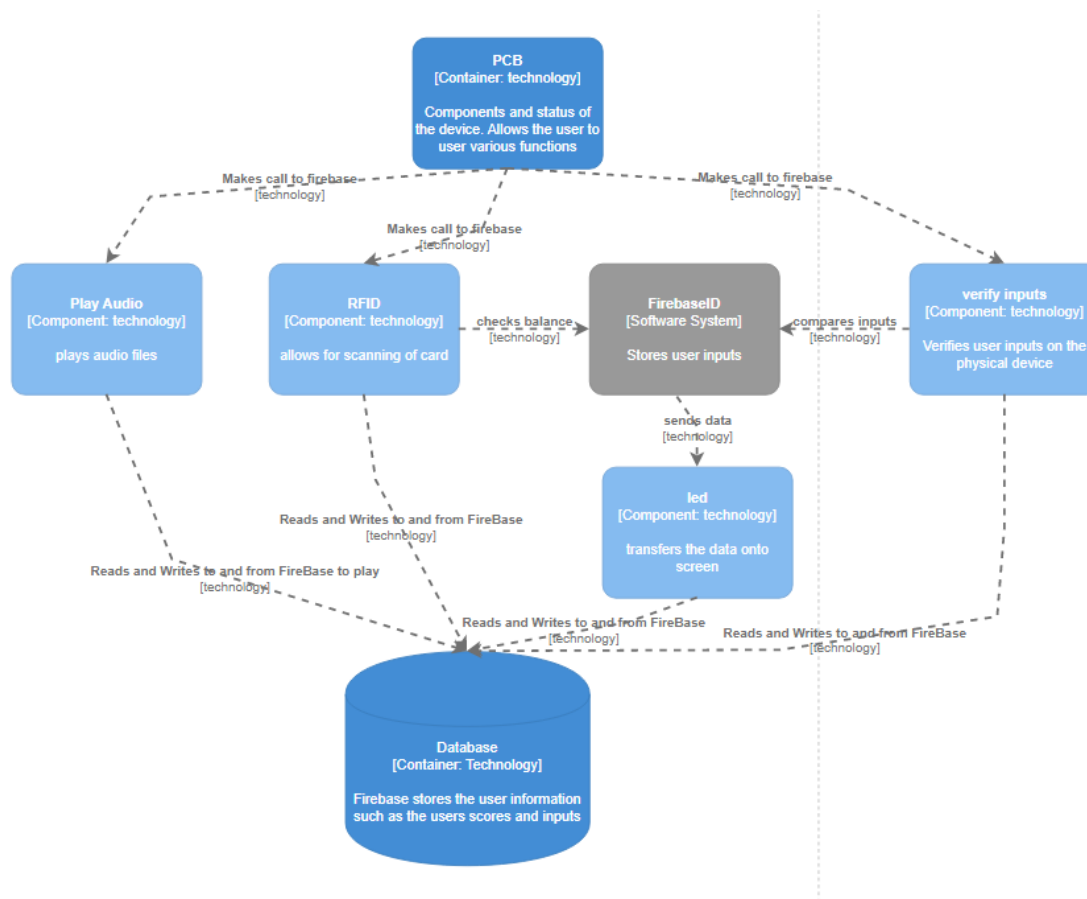


Component Diagrams

Component Diagram 1 (Mobile App)



Component Diagram 2 (PCB)



Areas of Refactoring

Area 1

Inside of Profile Fragment, I made a huge function that had multiple different objectives and tasks, this function `loadImage()` was responsible for loading the users profile image, check to see if the user is connected to the internet or not, and then obtain and display users data from firebase while also saving it to shared preference and also containing a functionality where if the function is used in offline mode, it would save the users data to shared preference to which if the user then went online, it would update the firebase with their shared preference score that was gained in offline mode. This code was a huge source of problems and my teammates were complaining to me that it wasn't good due to it being unreadable and hard to understand. It complicated the code further and it was bad for managing our time for what it did. That's why I refactored it. I redesigned it and split it up into much more manageable and understandable parts.

```

public void loadImage() {
    preferenceManager=PreferenceManager.getInstance(getActivity());

    SharedPreferences sharedPreferences= this.getActivity().getSharedPreferences(getString(R.string.SettingsPref),
Context.MODE_PRIVATE);
    SharedPreferences.Editor editor = sharedPreferences.edit();

    ConnectivityManager connectivityManager = (ConnectivityManager)
getActivity().getSystemService(Context.CONNECTIVITY_SERVICE);
    if(connectivityManager.getNetworkInfo(ConnectivityManager.TYPE_MOBILE).getState() ==
NetworkInfo.State.CONNECTED ||
        connectivityManager.getNetworkInfo(ConnectivityManager.TYPE_WIFI).getState() ==
NetworkInfo.State.CONNECTED) {
        connected = true;
    }
    else {
        connected = false;
    }
    if (connected == true) {

        if (sharedPreferences != null) {
            String typeOfLogin = sharedPreferences.getString("typeLogin", getString(R.string.blank));
            typeOfSignout = typeOfLogin;
            if (typeOfLogin.equals("GearAccount")) {
                String getUserId = sharedPreferences.getString(getString(R.string.userProfile),
getString(R.string.blank));
                //
                globalId = getUserId;
                //
                DatabaseReference reference =
FirebaseDatabase.getInstance().getReference(getString(R.string.childRef_reg_regFrag));
                Query checkUser =
reference.orderByChild(getString(R.string.childRef_username)).equalTo(getUserId);
                //
                DatabaseReference mDatabase = FirebaseDatabase.getInstance().getReference();
                checkUser.addListenerForSingleValueEvent(new ValueEventListener() {
                    @Override
                    public void onDataChange(@NonNull DataSnapshot snapshot) {
                        if (snapshot.exists()) {
                            String userName =
snapshot.child(getUserId).child(getString(R.string.childRef_username)).getValue(String.class);
                            Integer userScore =
snapshot.child(getUserId).child(getString(R.string.childRef_topScore)).getValue(Integer.class);
                            Integer userCur =
snapshot.child(getUserId).child(getString(R.string.childRef_Currency)).getValue(Integer.class);
                            String scoreStore = Integer.toString(userScore);
                            String curStore = Integer.toString(userCur);
                            editor.putString("profileUsername", userName);
                            editor.putString("profileScore", scoreStore);
                            editor.putString("profileCur", curStore);
                            editor.apply();

                            String loadName = sharedPreferences.getString("profileUsername", profileUser);
                            String loadCur = sharedPreferences.getString("profileCur", profileCur);
                            String loadScore = sharedPreferences.getString("profileScore", profileScore);
                            int offlineScore = sharedPreferences.getInt("offlineScore", profileScoreKeep);

                            usernameTextView.setText(getString(R.string.usernameDisplay) + loadName);
                            Log.d("Online OfflineScore", Integer.toString(offlineScore));
                            Log.d("Online loadScore", loadScore);

                            if (offlineScore > Integer.parseInt(loadScore)) {
                                topScoreTextView.setText(getString(R.string.scoreDisplay) + offlineScore);
                                DatabaseReference mDatabase = FirebaseDatabase.getInstance().getReference();
                                mDatabase.child(getString(R.string.childRef_reg_regFrag)).child(getUserID).child(getString(R.string.childRef_topScore))
                                ).setValue(offlineScore);
                            }
                        }
                    }
                });
            }
        }
    }
}

```

```

        } else {
            topScoreTextView.setText(getString(R.string.scoreDisplay) + loadScore);
        }

        currencyTextView.setText(getString(R.string.currencyDisplay) + loadCur +
getString(R.string.gears));

        } else {
            Log.d(getString(R.string.FAILED), "FAILED GEAR");
        }
    }

    @Override
    public void onCancelled(@NonNull DatabaseError error) {

    }

    });
} else if (typeOfLogin.equals("GearGoogleAccount")) {
    String getUserId = sharedPreferences.getString(getString(R.string.userProfile),
getString(R.string.blank));
    globalId = getUserId;
    DatabaseReference db = FirebaseDatabase.getInstance().getReference();
    DatabaseReference uidRef = db.child("users").child(getUserId);
    uidRef.get().addOnCompleteListener(new OnCompleteListener<DataSnapshot>() {
        @Override
        public void onComplete(@NonNull Task<DataSnapshot> task) {
            if (task.isSuccessful()) {
                DataSnapshot snapshot = task.getResult();
                String userName =
snapshot.child(getString(R.string.childRef_username)).getValue(String.class);
                Integer userScore =
snapshot.child(getString(R.string.childRef_topScore)).getValue(Integer.class);
                Integer userCur =
snapshot.child(getString(R.string.childRef_Currency)).getValue(Integer.class);
                String scoreStore = Integer.toString(userScore);
                String curStore = Integer.toString(userCur);
                editor.putString("profileUsername", userName);
                editor.putString("profileScore", scoreStore);
                editor.putString("profileCur", curStore);
                editor.apply();

                String loadName = sharedPreferences.getString("profileUsername", profileUser);
                String loadCur = sharedPreferences.getString("profileCur", profileCur);
                String loadScore = sharedPreferences.getString("profileScore", profileScore);
                int offlineScore = sharedPreferences.getInt("offlineScore", profileScoreKeep);

                usernameTextView.setText(getString(R.string.usernameDisplay) + loadName);
                Log.d("Online OfflineScore", Integer.toString(offlineScore));
                Log.d("Online loadScore", loadScore);

                if (offlineScore > Integer.parseInt(loadScore)) {
                    topScoreTextView.setText(getString(R.string.scoreDisplay) + offlineScore);
                    DatabaseReference mDatabase = FirebaseDatabase.getInstance().getReference();

mDatabase.child(getString(R.string.childRef_reg_regFrag)).child(getUserID).child(getString(R.string.childRef_topScore)
).setValue(offlineScore);

                    } else {
                        topScoreTextView.setText(getString(R.string.scoreDisplay) + loadScore);
                    }

                    currencyTextView.setText(getString(R.string.currencyDisplay) + loadCur +
getString(R.string.gears));
                } else {
                    Log.d(getString(R.string.FAILED), "FAILED GOOGLE LOAD PROF");
                }
            }
        }
    }
}

```

```

    });

    } else {
        Log.d(getString(R.string.FAILED), getString(R.string.FAILED));
    }

}

} else {
    String loadName = sharedPreferences.getString("profileUsername", profileUser);
    String loadCur = sharedPreferences.getString("profileCur", profileCur);
    String loadScore = sharedPreferences.getString("profileScore", profileScore);
    int offlineScore = sharedPreferences.getInt("offlineScore", profileScoreKeep);

    usernameTextView.setText(getString(R.string.usernameDisplay) + loadName);
    Log.d("Offline OfflineScore", Integer.toString(offlineScore));

    if (offlineScore > Integer.parseInt(loadScore)) {
        topScoreTextView.setText(getString(R.string.scoreDisplay) + offlineScore);
    } else {
        topScoreTextView.setText(getString(R.string.scoreDisplay) + loadScore);
    }

    currencyTextView.setText(getString(R.string.currencyDisplay) + loadCur + getString(R.string.gears));

}

String previouslyEncodedImage = preferenceManager.getString(getString(R.string.image_data));

if( !previouslyEncodedImage.equalsIgnoreCase(getString(R.string.blank)) ){
    byte[] b = Base64.decode(previouslyEncodedImage, Base64.DEFAULT);
    Bitmap bitmap = BitmapFactory.decodeByteArray(b, 0, b.length);
    mImageView.setImageBitmap(bitmap);
}

}

```

Area 2

The goal of this fragment within ScoreFragment was to present everyone's highest score and arrange it in descending order. The user will be seen on the phone's ListView. My issue was that whenever Firebase was updated, such as with a new score, the app and UI would crash. Displaying only the user's top scores rather than the overall results was the solution to this issue.

```

// CENG-322-0NC Francisco Santos n01423860, Pradeep Singh n00975892
// CENG-322-0NB Ralph Robes n01410324, Elijah Tanimowo n01433560
package ca.nika.it.gear5;

import android.app.ProgressDialog;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.os.AsyncTask;
import android.os.Bundle;

import androidx.annotation.NonNull;
import androidx.fragment.app.Fragment;

import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.ImageView;

```

```

import android.widget.ListView;
import android.widget.TextView;
import android.widget.Toast;

import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;

import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

import java.io.InputStream;
import java.net.URL;
import java.util.ArrayList;
import java.util.Collection;
import java.util.Collections;
import java.util.Comparator;
import java.util.List;
import java.util.concurrent.TimeUnit;

import ca.nika.it.gear5.LoginSetup.LoginFragment;

public class ScoreFragment extends Fragment {

    Button refreshBtn;

    ListView listView;

    ProgressDialog progress;
    ImageView nikaSyncTaskImage;

    public ScoreFragment() {

    }

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
    }

    private class LoadImage extends AsyncTask<Void,Void,Bitmap> {
        @Override
        protected Bitmap doInBackground(Void... params) {
            Bitmap bitmap=null;

            try {
                bitmap= BitmapFactory.decodeStream((InputStream)new URL(getString(R.string.onlineLink)).getContent());
            } catch (Exception e) {
                e.printStackTrace();
            }

            return bitmap;
        }

        @Override
        protected void onPreExecute() {
            //show progress dialog while image is loading
            progress=new ProgressDialog(getActivity());
            progress.setMessage(getString(R.string.loadingImage));
        }
    }

```



```

        progress.show();
    }

    @Override
    protected void onPostExecute(Bitmap bitmap) {
        if(bitmap!=null) {
            nikaSyncTaskImage.setImageBitmap(bitmap);
            progress.dismiss();
        } else {
            progress.dismiss();
            Toast.makeText(getActivity(), getString(R.string.errorOccured), Toast.LENGTH_LONG).show();
        }
    }
}

@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container,
                           Bundle savedInstanceState) {
    View view = inflater.inflate(R.layout.fragment_score, container, false);

    listView = (ListView) view.findViewById(R.id.nika_userList);
    nikaSyncTaskImage = (ImageView) view.findViewById(R.id.nika_aSync);
    refreshBtn = (Button) view.findViewById(R.id.refresh_btn);

    refreshBtn.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {

            JSONArray arrayFirebase=new JSONArray();
            JSONArray sortedArray=new JSONArray();
            List<JSONObject> sortValues = new ArrayList<JSONObject>();

            FirebaseDatabase database = FirebaseDatabase.getInstance();
            DatabaseReference myRef = database.getReference(getString(R.string.childRef_reg_regFrag));
            myRef.addValueEventListener(new ValueEventListener() {
                @Override
                public void onDataChange(DataSnapshot snapshot) {
                    for (DataSnapshot item : snapshot.getChildren()){
                        String username = item.child(getString(R.string.childRef_username)).getValue().toString();
                        String userScore =
item.child(getString(R.string.childRef_topScore)).getValue(Integer.class).toString();
                        try {
                            arrayFirebase.put(new JSONObject().put("Username", username).put("UserScore",
userScore));
                        } catch (JSONException e) {
                            Log.d("Failed", e.toString());
                        }
                    }

                    for (int i = 0; i < arrayFirebase.length(); i++){
                        try {
                            sortValues.add(arrayFirebase.getJSONObject(i));
                        } catch (JSONException e) {
                            e.printStackTrace();
                        }
                    }

                    Collections.sort(sortValues, new Comparator<JSONObject>() {
                        private static final String KEY_NAME = "UserScore";
                        @Override
                        public int compare(JSONObject a, JSONObject b) {
                            String str1 = new String();
                            String str2 = new String();

                            try {

```

```

        str1 = (String) a.get(KEY_NAME);
        str2 = (String) b.get(KEY_NAME);
    } catch (JSONException e) {
        Log.d("Failed", e.toString());
    }
    return -str1.compareTo(str2);
}
});

for(int i = 0; i < arrayFirebase.length(); i++) {
    sortedArray.put(sortValues.get(i));
}

try {
    DisplayTopUsers(sortedArray);
} catch (JSONException e) {
    Log.d("Failed", e.toString());
}

}

@Override
public void onCancelled(@NonNull DatabaseError error) {

}

});

}

});

new LoadImage().execute();

return view;

}

private void DisplayTopUsers(JSONArray sortedArray) throws JSONException {
    ArrayList<String> top15Array = new ArrayList<String>();

    if (sortedArray.length() < 15) {
        for (int pos = 0; pos < sortedArray.length(); pos++) {
            String jsonStr = sortedArray.getString(pos);
            JSONObject objectData = new JSONObject(jsonStr);
            String Score = (String) objectData.get("UserScore");
            String Username = (String) objectData.get("Username");
            String combinedData = Username + " : " + Score;
            top15Array.add(combinedData);
        }
    } else {
        for (int pos = 0; pos < 15; pos++) {
            String jsonStr = sortedArray.getString(pos);
            JSONObject objectData = new JSONObject(jsonStr);
            String Score = (String) objectData.get("UserScore");
            String Username = (String) objectData.get("Username");
            String combinedData = Username + " score: " + Score;
            top15Array.add(combinedData);
        }
    }
    ArrayAdapter adapter = new ArrayAdapter<String>(getActivity(),
        R.layout.list_view, top15Array);

```

```
listView.setAdapter(adapter);
```

```
}
```

```
}
```